CSEP 561 – Retransmissions (ARQ)

David Wetherall
djw@cs.washington.edu
We want to send messages reliably. We’ve seen codes to correct and detect errors. Next …

Retransmissions are used in combination with coding

- Message resend for all types of “loss”
- At different layers, but we’re mostly thinking about links for now

IP doesn’t retransmit
Reliability strategies

To send messages despite errors, we can:

1. Use error-correcting code
   - Sometimes called Forward Error Correction (FEC)
2. Use error-detecting codes plus retransmissions
   - Retransmit the frames received in error
3. Use a combination of the above
   - Correct for expected error, detect and repair remaining errors

• Note: typical to “throw away” a message if uncorrected errors are detected. This converts errors to the broader category of message loss
Retransmissions, or more formally
Automatic Repeat Request (ARQ)

- Sender automatically resends after a timeout until a positive acknowledgment (ACK) is obtained from the receiver
- Receiver automatically acknowledges frames (packets) that are not corrupted or lost in the network
- ARQ is generic name for protocols based on this strategy
Two non-trivial issues

1. How long to set the timeout?
   – Issue is timing variability, e.g., processing delays
   – Way too long lowers performance, but otherwise some early timeouts
   – Easy for direct links (e.g., 802.11, set an upper bound)
   – Harder for networks → we will return to this w/ TCP

2. How to avoid accepting duplicate frames as new
   – Given retransmissions, frame loss, and imprecise timeouts

Want performance in the common case, correctness always
The Need for Sequence Numbers

- In the case of ACK loss (or poor choice of timeout) the receiver can’t distinguish current message from next
- Solution: put sequence number on frames and ACKs
Stop-and-Wait

- Only one outstanding frame at a time, 0 or 1.
- Retransmissions re-sent with same number
- Number only needs to distinguish between current and next frame
  - A single bit will do
Limitation of Stop-and-Wait

- Lousy performance if wire time $<<$ prop. delay
  - How bad? You do the math
- Want to utilize all available bandwidth
  - Need to keep more data “in flight”
  - How much? Remember the bandwidth-delay product?
- Leads to Sliding Window Protocol (later)
  - Used by TCP, and good for long links (e.g., satellite)
  - But for local wireless (802.11, cellular), stop-and-wait will do
When to use ARQ or FEC?

- Will depend on the kind of errors and cost of recovery
- Example: Message with 1000 bits, Prob(bit error) 0.001
  - Case 1: random errors
  - Case 2: bursts of 1000 errors

- Q: What to use in Case 1 and 2?
ARQ vs. FEC

• FEC used at low-level to lower residual error rate
• ARQ often used to fix large errors, e.g., packet collision, and with detection to protect against residual errors

• FEC sometimes used at high level too:
  – Real time applications (no time to retransmit!)
  – Nice interaction with broadcast (different receiver errors!)
Example: 802.11 wireless

The standard design is:

- **PHY**: FEC on data via a binary convolutional code or LDPC, with interleaving for burst errors
  - Rates from ½ to 5/6.
  - Symbol errors are common

- **Link**: 32-bit CRC on frame with stop-and-wait
  - Retransmissions are common
Example: wired Ethernet

- Many versions, consider gigabit Ethernet

- PHY: some FEC is built into modulation scheme (2 bits in 5 voltage levels)

- Link: 32-bit CRC on frame with no retransmissions.